APPLICATION UNDER UNITED STATES PATENT LAWS

Atty. Dkt. No. PW 264796
(M#)

Invention: TREE-STYLE HIERARCHICAL CONTROL WITH GRAPHICAL DEPICTION OF NON-HIERARCHICAL INTERRELATIONSHIPS

Inventor (s): MCCLELLAN, James R.

Cognex Corporation
One Vision Drive
Natick
MA. 01760-2059
Telephone: (508) 650-3039

This is a:

- ☐ Provisional Application
- ☒ Regular Utility Application
- ☐ Continuing Application
  - ☒ The contents of the parent are incorporated by reference
- ☐ PCT National Phase Application
- ☐ Design Application
- ☐ Reissue Application
- ☐ Plant Application
- ☐ Substitute Specification
  Sub. Spec Filed _____
  in App. No. ___/_____
- ☐ Marked up Specification re
  Sub. Spec. filed _____
  In App. No ___/_____

SPECIFICATION

TREE-STYLE HIERARCHICAL CONTROL WITH GRAPHICAL DEPICTION OF NON-HIERARCHICAL INTERRELATIONSHIPS

1. Prior Application Data

2. Reservation of Copyright

BACKGROUND

3. Field of the Invention

The present invention, in certain respects, relates to the field of graphical user interface. In other respects, the present invention relates to a method and system that creates a tree-style graphical representation that depicts interrelationships among a set of entities.

4. Description of Background Information

A tree-style representation can be used to describe hierarchical relationships. An example of a hierarchical relationship is a parent-child relationship. Entities connected by hierarchical relationships form a hierarchy, which can be depicted using a tree representation. For instance, directories in a computer file system form a hierarchy which can be unambiguously described by a tree representation.

A tree representation can be visualized in different ways. For example, each parent-child relationship can be expressed using an arrow linking the parent and the child, with arrow pointing at the child. Alternatively, left-indentation is also frequently used to visualize a hierarchical relationship in a tree representation. Fig. 1 illustrates an exemplary tree-style graphical representation for a directory hierarchy of a computer file system, displayed, as a graphical user interface, on a computer monitor. In Fig. 1, the subdirectory in a hierarchical relationship (e.g., directory vs. subdirectory) appears left-indented from where its parent directory appears. For example, "FreeLane" in Fig. 1 is a subdirectory of "Program Files" and it is left-indented from where "Program Files" is displayed in the interface. In general, a tree representation can be used to describe hierarchical relationship among a set of entities. Such entities can be people (e.g., parents and children), directories on a computer file system, computer programs, organizations of government, and categories of books.

As shown in Fig. 1, the hierarchical relationships among different directories can be readily seen through a tree-style graphical representation. In addition, such a tree-style graphical representation also provides a means for a user to interactively manipulate the underlying hierarchy. For example, a directory in the hierarchy can be selected, for deletion, by simply clicking on the node that represents the directory. Therefore, a tree is an effective representation for hierarchical relationships. A tree-style graphical representation provides a useful means to manipulate or control a hierarchy.

There are non-hierarchical relationships. For example, data flow relation is not hierarchical. Different representations can be used to describe non-hierarchical relationships. For example, a block diagram can be used to depict data flow relationships among different functional units. Fig. 2 shows an exemplary block diagram, in which data flows between different functional units are described as the links between the functional

blocks. For example, "2D Images" is flown from Image Acquisition Unit (a functional unit), where 2D images are acquired, to Image Processing Unit (another functional unit), where acquired 2D images may be processed.

The above tree representations do not depict non-hierarchical relationships such as the data flow among different entities within the hierarchy. Although existing block diagram techniques do show non-hierarchical relationships among entities, they do not depict the hierarchical relationships of those entities.

SUMMARY OF THE INVENTION

The present invention is provided to improve a tree-style representation to depict relationships among a set of entities. More specifically, an improved method and system is presented that addresses the generation and modification of a tree-style graphical representation that allows simultaneous depiction of both hierarchical and non-hierarchical interrelationships among a set of entities.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is further described in the detailed description which follows, by reference to the noted drawings by way of non-limiting exemplary embodiments, in which like reference numerals represent similar parts throughout the several views of the drawings, and wherein:

Fig. 1 illustrates an exemplary tree representation for a directory hierarchy in a computer system;

Fig. 2 illustrates an exemplary block diagram that describes data flow relationships among different functional units;

Fig. 3A shows an example of a tree-style graphical representation that depict both the hierarchical interrelationships among a set of machine vision related tools as well as the non-hierarchical data flow interrelationships among those tools.

Fig. 3 illustrates an exemplary embodiment of the present invention, in which a tree-style graphical representation can be generated and modified;

Fig. 4 shows an exemplary high level block diagram for an embodiment of the present invention, in which specifications for interrelationships can be acquired and stored;

Fig. 5 is an exemplary flowchart of a process to acquire interrelationship specifications;

Fig. 6 illustrates an exemplary high level block diagram for an embodiment of the present invention, in which a tree-style graphical representation is generated based on interrelationship specifications;

Fig. 7 is an exemplary flowchart of a process to generate a tree-style graphical representation;

Fig. 8 is an exemplary flowchart of a process to add an entity to an existing tree-style graphical representation;

Fig. 9 is an exemplary flowchart of a process to delete an entity from an existing tree-style graphical representation;

Fig. 10 is an exemplary flowchart of a process to add an interrelationship to an existing tree-style graphical representation;

Fig. 11 is an exemplary flowchart of a process to delete an interrelationship from an existing tree-style graphical representation; and

Fig. 12 is an exemplary flowchart of a process to update an interrelationship in an existing tree-style graphical representation.

DETAILED DESCRIPTION

There is a need for a tree-style graphical representation that depicts both hierarchical and non-hierarchical relationships among a set of entities. Fig. 3A illustrates an example of such a graphical representation depicting the hierarchical and non-hierarchical interrelationships among a set of machine vision related entities. In this example, a left-indentation method is used to construct a tree depicting hierarchical relationships (e.g., OutputImage under ImageDatabased is left-indented). Links on the right side of the tree describe non-hierarchical relationships (e.g., the arrows coming out of OutputImage under Imagedatabase pointing to various entities appear on the right side of the left-indented tree).

In Fig. 3A, there are five entities at the same level of the hierarchy (ImageDatabase, Caliper, PMAligh, Fixture, and LightMeter). Each of those entities has lower level entities. The left-indented tree (the left side of the representation shown in Fig. 3A) alone does not indicate any relationship between, for example, ImageDatabase and Caliper even though these two entities may be related through a data flow relationship. For example, Caliper may retrieve an image from ImageDatabased as input (i.e., OutputImage of ImageDatabase is the InputImage of Caliper). PMAlign may also retrieve both an image as well as a pattern image from ImageDatabase as input and then produces PoseInformation as its output. Such pose information may be further used by Fixture to apply to an image, retrieved from ImageDatabase, to generate an output image. Such data flow relationships are non-hierarchical and are clearly represented in Fig. 3A. An embodiment of the invention is presented that enables the generation of such a tree-style graphical representation.

The processing described below may be performed by a general-purpose computer alone or in connection with a specialized computer. Such processing and functionality can be implemented in the form of special purpose hardware or in the form of software being run by a general-purpose computer. Any data handled in such processing or created as a result of such processing can be stored in any memory as is conventional in the art. By way of example, such data may be stored in a temporary memory, such as in the RAM of a given computer system or subsystem. In addition, or in the alternative, such data may be stored in longer-term storage devices, for example, magnetic disks, rewritable optical disks, and so on. For purposes of the disclosure herein, a computer-readable media may comprise any form of data storage mechanism, including such existing memory technologies as well as hardware or circuit representations of such structures and of such data.

Fig. 3 shows a system 100 that generates a tree-style graphical representation to depict both hierarchical and non-hierarchical interrelationships among a set of entities. The illustrated system 100 comprises a tree-style graphical representation generation unit 140, an acquisition mechanism 130, a storage mechanism 160, and a display unit 120.

Acquisition mechanism 130, in the illustrated embodiment, is for acquiring the specifications of entities and the hierarchical and non-hierarchical interrelationships among these entities. It may comprise different approaches to obtaining the specifications. For example, acquisition mechanism 130 may load a digital file containing the specifications and then parse the file to extract the specifications. Acquisition mechanism 130 may also obtain the specifications interactively via a graphical user interface.

Specifications acquired by acquisition mechanism 130 may comprise the information that defines a set of entities and the information that describes the interrelationships among those entities. Interrelationships may include both hierarchical as

well as non-hierarchical interrelationships. For example, in an object-oriented computer program, objects and their properties form hierarchical relationships. In this case, objects and their properties may be specified as entities. Each object and its properties may be connected by hierarchical relationships. An example of a non-hierarchical interrelationship in an object-oriented computer system may be the data flow relationships between objects.

Specifications acquired by acquisition mechanism 130 are stored in storage mechanism 160. Storage mechanism 160 may retain specifications of hierarchical interrelationships separately from where specifications of non-hierarchical interrelationships are stored so that different types of interrelationship specifications may be maintained and retrieved individually. Storage mechanism 160 may also comprise a separate storage 270 for tree representations that are generated based on the stored specifications.

In Fig. 3, tree-style graphical representation generation unit 140 retrieves specifications stored in storage mechanism 160 and then generates a tree-style graphical representation that may depict both hierarchical interrelationships and non-hierarchical interrelationships, according to the specifications.

Tree-style graphical representation generation unit 140, in the illustrated embodiment, may comprise different processing components. For example, one processing component may generate an initial tree representation based on the specifications of hierarchical interrelationships. Based on the initial tree representation, a different component may generate an augmented tree representation that incorporates non-hierarchical interrelationships in the initial tree representation, according to the specifications of non-hierarchical interrelationships. Yet another different component

may display the augmented tree representation, as a graphical user interface, on display unit 120.

A tree representation, constructed from specifications of entities and interrelationships, may be implemented in an abstract form such as a data structure. Such an abstract tree representation is rendered into its graphical form, or a tree-style graphical representation, before it can be displayed graphically on a display device.

Display unit 120 in Fig. 3 may comprise a screen, as a display device, and a control device that controls the display device. For example, the display device may be a computer monitor. The control device may comprise a personal computer, a keyboard, and a pointing device such as a mouse. A tree-style graphical representation, generated by tree-style graphical representation generation unit 140, may be graphically rendered and displayed on a display device, included in display unit 120. An interactive graphical interface may also be displayed and functioned on the same display device of display unit 120 so that acquisition mechanism 130 may interactively acquire specifications about entities and interrelationships of the entities in a graphical means.

Fig. 4 shows a more detailed system configuration for acquiring specifications of interrelationships. In particular, Fig. 4 shows how acquisition mechanism 130 relates to display unit 120 and storage mechanism 160. In fig. 4, acquisition mechanism 130 comprises, for example, two components: a batch extractor 230 and an interactive extractor 240. Storage mechanism 160 comprises, for example, three parts: part 250 to store hierarchical interrelationship specifications, part 260 to store non-hierarchical interrelationship specifications, and part 270 to store tree representations.

Batch extractor 230 extracts specifications of entities and interrelationships in a batch mode from, for example, a digital file. The digital file may be stored at a remote site on a computer-readable medium 210, such as a disk or a CD-ROM, and may be accessed

across network. Batch extractor 230 reads a digital file and extracts specifications from the digital file by parsing the file. Specifications about entities, hierarchical interrelationships, and non-hierarchical interrelationships may be extracted individually as different sets of specifications. Such identified individual sets of specifications may then be stored in corresponding parts of storage mechanism 160. For example, the set of specifications that describe hierarchical interrelationships may be stored in part 250 of storage mechanism 160. The set of specifications that describe non-hierarchical interrelationships may be stored in part 260 of storage mechanism 160.

Acquisition mechanism 130 may also operate in a different mode such as an interactive mode. In this case, interactive extractor 240 interacts with an interactive graphical interface 220, through which definitions of entities as well as specifications about their interrelationships may be obtained in an interactive fashion via a, for example, graphical means. The interactive graphical user interface 220 may be displayed on a display device of display unit 120 and driven by interactive extractor 220 through a control device of display unit 120.

An exemplary flowchart of the process of interactively acquiring specifications is illustrated in Fig. 5. For example, a set of entities may be defined first at act 310 in an interactive session, in which the names of the entities may be entered through a graphical user interface. An interrelationship may be defined in a subsequent interactive session in, for example, the following sequence of acts. The entities are defined and displayed first in the graphical user interface at act 310 so that they can be graphically selected. A user may then define an interrelationship between two entities by interactively providing a description of the interrelationship and the associated two entities.

The two entities that form an interrelationship are selected at acts 320 and 330, respectively. The selection, made at acts 320 and 330, may be accomplished by simply,

for example, clicking on the two entities displayed in the user interface. The description about an interrelationship between the two selected entities is obtained at act 340 from a user interface. The description may include information about the name of the interrelationship, the type of relation (e.g., hierarchical or non-hierarchical), the role each entity plays in the relation (e.g., one entity is the child and one entity is the parent in a hierarchical interrelationship), or the annotation of the relation.

The specifications acquired by interactive extractor 240 are stored at act 350 in corresponding parts of storage mechanism 160. For example, specifications of hierarchical interrelationships may be stored in part 250. Specifications of non-hierarchical interrelationships may be stored in part 260.

A tree-style graphical representation can be constructed based on acquired specifications. Fig. 6 shows an exemplary high level block diagram for generating a tree-style graphical representation. In Fig. 6, the relationships among storage mechanism 160, tree-style graphical representation generation unit 140, and display unit 120 are shown. Tree-style graphical representation generation unit 140 comprises, for example, three components: an initial tree representation generator 430, a augmented tree generator 440, and a rendering unit 450.

Initial tree representation generator 430 retrieves, as illustrated in the exemplary diagram shown in Fig. 6, specifications of hierarchical interrelationships from storage mechanism 160. Based on the input specifications, initial tree generator 430 generates an initial tree representation that depicts only the hierarchical interrelationships described by the input specifications. The initial tree representation is then fed into augmented tree generator 440. Augmented tree generator 440 retrieves the specifications for non-hierarchical interrelationships for the same set of entities from part 260 of storage mechanism 160. Based on the specifications for non-hierarchical interrelationships,

augmented tree generator 440 depicts the non-hierarchical interrelationships in the initial tree representation. It should be noted that the depiction of non-hierarchical interrelationships does not alter the initial tree structure formed with respect to the hierarchical interrelationships. The augmented tree simultaneously depicts the structure of the hierarchy, described by the hierarchical relationships, and the non-hierarchical relationships among the entities within the hierarchy.

An augmented tree representation, once generated, may be stored in part 270 of storage mechanism 160. The tree representation generated at this stage may be implemented in its abstract form, such as a data structure, instead of in its graphical form. Rendering unit 450 takes an augmented tree representation and renders the tree representation into its graphical form to generate a tree-style graphical representation and displays the graphical representation on display unit 120.

Rendering unit 450 may also graphically convey information about interrelationships in the representation. In an exemplary implementation, line styles or colors are associated with particular kinds of relationships. For instance, a dashed line style may indicate that an interrelationship is invalid. Further, various colors may correspond to types of data flow, such as, for example, images, coordinate transforms, and simple numbers. Rendering unit 450 may also dynamically vary styles or colors for the benefit of the user. In particular, when an entity is selected, all lines linked thereto may become wider so that the user may better visualize the interrelationships of that node.

Rendering unit 450 may provide so-called "tool tips" to convey additional information about the nature of relationships. When a user rests a pointing device over a link, a small pop-up window appears that describes the link. For instance, the window may describe the data type associated with the link.

Fig. 3A shows minus signs to the left of the ImageDatabase, Caliper, PMAlign, Fixture, and LightMeter entities. The respective sub-entities may be hidden by clicking on each minus sign. When sub-entities are hidden, links may not be drawn to the sub-entities. However, the node or nodes to which the sub-entities are linked may be visible. In an exemplary implementation, rendering unit 450 may graphically depict links to hidden sub-entities. For instance, rendering unit 450 may display the symbol "|---->" adjacent to visible entities. Such a symbol may be rendered in various colors and may be associated with pop-up windows. As such, additional information about the link may be conveyed to a user.

Fig. 7 shows an exemplary flowchart of a process of generating a tree-style graphical representation. Specifications of hierarchical interrelationships are obtained at act 510 first. Based on the specifications, initial tree representation generator 430 constructs an initial tree representation at 520. Taking the initial tree representation as input, augmented tree generator 440 retrieves specifications of non-hierarchical interrelationships at act 530. Based on the specifications of non-hierarchical interrelationships, augmented tree generator 440 incorporates the non-hierarchical interrelationships in the initial tree representation to generate an augmented tree representation at act 540. The augmented tree representation is then stored at act 550 in part 270 of storage mechanism 160. Finally, rendering unit 450 renders the augmented tree representation and displays it at act 560 on display unit 120.

A tree-style graphical representation, displayed on a display device, may be modified. Such modification comprises, for example, adding an entity, deleting an entity, adding an interrelationship, deleting an interrelationship, and updating an interrelationship. System 100 shown in Fig. 3 may also be used to modify a tree-style graphical

representation, in a similar fashion as a tree is generated, via either a non-graphical or a graphical means.

Modifications to a tree-style representation may be described by a modification specification. For example, if a modification involves adding an entity to a tree, a modification specification may provide information about the new entity and the location in the hierarchy that the new entity may be inserted. Specifically, a modification specification that describes to add an hierarchical interrelationship to a tree may include, for example, the definition of two entities, one defined as the parent entity and the other defined as the child entity, and the definition of the hierarchical relationship to be added such as the name of the relation and characteristics of the relation.

A modification specification may be stored in a digital file on a computer-readable medium, such as medium 210 in Fig. 4, and can be read by batch extractor 230. In this case, one or more modification specifications may be specified in a single digital file. The modifications specified in the digital file may be carried out all at once in a batch mode.

A modification specification may also be acquired interactively, by interactive extractor 240, via interactive graphical user interface 220. In this case, modifications to a tree-style graphical representation may be performed interactively via a graphical means. Fig. 8 to Fig. 12 present exemplary flowcharts of processes that interactively perform different types of modification to a tree-style graphical representation.

Fig. 8 is an exemplary flowchart of a process that adds a new entity to an existing tree-style graphical representation. The existing tree representation is first displayed in a graphical user interface at act 610. A pointing device may be activated in the graphical user interface. The new entity is defined at act 620 by, for example, interactively entering, through the graphical user interface, the name of the new entity as well as other associated information. The location where the new entity will be inserted in the tree is specified at

act 630. The location to insert the new entity specified at act 630 may be a simple pointing and clicking (at the insertion position in the existing tree) using a mouse connected to the display unit and to the graphical user interface.

Once the new entity is defined and the insertion location is specified, the new entity is added to the existing tree at act 640. The act performed at 640 may involve both updating the abstract representation of the existing tree, such as a data structure, as well as updating the graphical representation of the existing tree on a screen. The updated abstract tree representation is stored in storage mechanism 160 at act 650 and the updated graphical representation is displayed at act 660.

Fig. 9 shows a flowchart of a process that deletes an existing entity from a tree-style graphical representation. The tree to be modified is displayed first in a graphical user interface at act 710. From the displayed tree, a user may graphically select, at act 720, the entity to be deleted. The selection performed at act 720 may be accomplished by simply clicking on the entity in the graphical user interface using a device such as a mouse.

When an entity is to be deleted from a tree, all the interrelationships, both hierarchical and non-hierarchical, associated with the entity may have to be deleted accordingly. The interrelationships associated with the entity, selected to be deleted, are identified at act 730. Such associated interrelationships may be identified automatically based on, for example, the interrelationship specifications of the existing tree representation. Once identified, the associated interrelationships may be removed at act 740. The removal may include removing both corresponding specifications of the associated interrelationships from storage mechanism 160, as well as corresponding interrelationships from the tree representation.

The entity to be deleted is removed from the existing tree at act 750. A modified tree representation is then generated at act 760, which includes the modification to both

the abstract representation, such as a data structure, and the graphical representation of the existing tree. The modified tree representation is stored in part 270 of storage mechanism 160, either to replace the original tree representation or to be saved as a different version. Finally, the modified graphical representation of the tree is rendered at act 780 in the graphical user interface where the previous tree is displayed.

Interrelationships in a tree can also be modified interactively. Fig. 10 is an exemplary flowchart of a process that allows a user to interactively add an interrelationship to an existing tree. An existing tree is displayed first at act 810 in a graphical user interface. The interrelationship to be added is specified at act 820. The specification may include the information that define the entities that form the relationship, and the information that describes the type of the relationship (hierarchical or non-hierarchical), the characteristic of the relationship (e.g., uni-directional data flow), the role that each entity plays in the relationship (e.g., which entity supplies the data in a data flow relationship), and other descriptive information associated with the relation (e.g., the nature of the data flown from one entity to another in a data flow relationship).

Different types of information in the specification, acquired at act 820, may be obtained through different means. For example, a user may select two entities that form a new interrelationship by simply clicking on the entities in the existing tree displayed in the graphical user interface. To specify the type of the new interrelationship, the user may simply click on one of the buttons (displayed, for example, in the user interface to represent different types of interrelationships) that corresponds to the desired interrelationship type. Similarly, a user may specify the characteristics of the new relationship by selecting corresponding options, listed in a menu (e.g., a pull-down menu) associated with the button representing the chosen type of interrelationship.

For example, each button representing a different type of interrelationship may associate with a menu in which different characteristics of the corresponding type of interrelationship may be presented. For instance, a button representing a non-hierarchical interrelationship may associate with a menu that lists possible characteristics of a non-hierarchical interrelationship. Such characteristics may include non-directional, uni-directional, or bi-directional. A combination of a button selection and a menu selection, in this example, may represent one of non-directional non-hierarchical interrelationship, uni-directional non-hierarchical interrelationship, and bi-directional non-hierarchical interrelationship, respectively.

A user may also create interrelationships by dragging and dropping nodes. Specifically, the user may select one or more nodes and drag them onto another node to create a desired relationship among the nodes. Graphical feedback may be conveyed to the user during the dragging process. For instance, when a selected node is dragged over a potential destination node, a rendering unit may draw all links that will be created if the selected node is dropped. All other links may be hidden during the dragging process. If the node is dropped, the rendering unit refreshes the display to depict the newly created relationship, as well as other relationships unaffected by the dragging and dropping process.

Other information about a new interrelationship, such as annotation of the relation, may be further acquired through the user interface. For example, a user may annotate a new bi-directional non-hierarchical interrelationship by entering the annotation, in a dialog box, that further specifies that the new interrelationship is a data-flow relationship between two entities and the data flows in both directions.

Using acquired modification specifications, the act at 830 produces a modified tree-style graphical representation in which the new interrelationship is incorporated to the

existing tree. Both the specifications of the new interrelationship, acquired at act 820, and the modified tree-style graphical representation, produced at act 830, are used to update, at act 840, the corresponding content stored in storage mechanism 160. For example, the specifications may be added to certain part of storage mechanism 160, depending on the type of the new interrelationship (if hierarchical, add to part 250, if non-hierarchical, add to part 260). Finally, the modified tree-style graphical representation is displayed in the graphical user interface at act 850 to replace the tree representing the original tree.

Fig. 11 shows an exemplary flowchart of a process that allows a user to interactively delete an interrelationship from an existing tree. The existing tree is first displayed at act 910 in a graphical user interface. A user may select the interrelationship to be removed at act 920 via a graphical means by, for example, clicking on the graphical rendering of the interrelationship. At act 930, the selected interrelationship is removed from the existing tree to generate a modified tree representation. The removal of the selected interrelationship, performed at act 930, may include removing both the corresponding specification for the interrelationship to be deleted from storage mechanism 160, as well as the corresponding interrelationship from the existing tree representation. The modified tree representation is stored, at act 940, in part 270 of storage mechanism 160, either to replace the original tree representation or to be saved as a different version. Finally, the modified tree is rendered at act 950 and to be displayed in the graphical user interface.

Fig. 12 shows an exemplary flowchart of a process that enables a user to interactively update an existing interrelationship in a tree. Updating an interrelationship may include changing the type (hierarchical to non-hierarchical), the characteristics (uni-directional to bi-directional), and the annotation (name of the data flown from one entity to another entity) of an existing interrelationship. Some type of update, such as changing

from non-hierarchical to hierarchical, may cause substantial changes in the corresponding tree representation. Other type of update, such as changing the annotation of an interrelation, may cause minimal change in the corresponding tree representation.

In Fig. 12, an existing tree is first displayed at act 1010 in a graphical user interface. Through this graphical user interface, a user may select, at act 1020, an interrelationship for updating by, for example, clicking on the graphical rendering of the interrelationship. To update an interrelationship, modification specification may need to be acquired first. The modification may then be carried out according to the modification specification.

In Fig. 12, a user may revise the existing specification of the selected interrelationship at act 1030 to produce a modification specification. This may be accomplished in a manner similar to, for example, what is described, referring to Fig. 10, for obtaining a specification of a new interrelationship. Based on the modification specification, the existing tree representation is revised at act 1040 to produce a modified tree-style representation.

The storage mechanism 160 and the tree graphical display also need to be accordingly updated. The modification specification for the modified interrelationship may replace the corresponding original specification stored in certain part of storage mechanism 160, depending on the type of interrelationship. The modified tree representation may replace the corresponding original tree representation, stored in part 270 of storage mechanism 160. The act at 1050 performs both types of update. The modified tree-style graphical representation is finally displayed at act 1060.

While the invention has been described with reference to certain illustrated embodiments, the words that have been used herein are words of description, rather than words of limitation. Changes may be made, within the purview of the appended claims,

without departing from the scope and spirit of the invention in its aspects. Although the invention has been described herein with reference to particular structures, acts, and materials, the invention is not to be limited to the particulars disclosed, but rather extends to all equivalent structures, acts, and materials, such as are within the scope of the appended claims.